COMP 302 Winter 2025 Problem Set 3

Define

```
type 'a susp = Susp of (unit -> 'a)
let force (Susp f) = f ()

type 'a stream = { hd: 'a; tl: 'a stream susp }
```

Problem 1: Power Series

Create a lazy stream representing the power series $\sum_{i=0}^{\infty} \frac{1}{2^i}$.

```
let rec power_series index x =
    (* Implement *)
```

Problem 2: Infinite List

Convert a regular list to an infinite list by repeating its elements indefinitely

```

cycle : 'a list -> 'a stream

let rec cycle list =
    (* Implement *)
```

Problem 3: Triple Fibonacci Sequence Generator

Create a function that generates a "Triple Fibonacci" sequence, where each term is the sum of the last three terms, starting with initial values.

```
let rec triple_fib a b c =
    (* Implement *)
```

Problem 4: Sorted Merge

Given two streams each containing integers, merge them into a single sorted stream without duplicates.

```ocaml
let rec merge_sorted s1 s2 =
  (* Implement * )
```

Problem 5: Change Making

Given the following implementation of the change-making problem using continuations in OCaml

```ocaml
(* Semi-CPS change-making with an accumulator *)
let rec change coins amt acc fail = match coins, amt with
  | _, 0 -> acc
  | [], _ -> fail ()
  | c::cs, amt when amt >= c ->
      change (c::cs) (amt-c) (c::acc) (fun () -> change cs amt acc
    fail)
  | c::cs, amt ->
      change cs amt acc fail
```

Transform this backtracking solution into a lazy stream generator that produces all possible ways to make change.

```ocaml
type 'a susp = Susp of (unit -> 'a)
type 'a lazy_list = { hd : 'a; tl : 'a fin_list susp }
and 'a fin_list = Empty | NonEmpty of 'a lazy_list

let rec go_gen_change coins amt acc next =
    (*Implement*)

let gen_change coins amt : int list fin_list =
    (*Implement* )
```